Model Checking the Remote Agent Planner (extended abstract)

Lina Khatib, Nicola Muscettola, and Klaus Havelund
NASA Ames Research Center, MS 269-2
Moffett Field, CA 94035
{*lina, mus, havelund*}@ptolemy.arc.nasa.gov

November 16, 2000

Keywords: Model Checking, Verification, Autonomy, Planning, Scheduling

This work tackles the problem of using Model Checking for the purpose of verifying the HSTS planning system. HSTS is the planner and scheduler of the remote agent autonomous control system deployed in Deep Space One (DS1)[8]. Model Checking allows for the verification of domain models as well as planning engines. We have chosen the real-time model checker UPPAAL for this work [6]. We start by motivating our work in the introduction. Then we give a brief description of HSTS and UPPAAL. After that, we give a sketch for the mapping of HSTS models into UPPAAL and we present samples of plan model properties one may want to verify.

1 Introduction

AI technologies, and specifically AI planning, facilitates the elicitation and automatic manipulation of system level constraints. However, the models used by the planner still need to be verified, i.e., it is necessary to guarantee that no unintended consequences will arise. One question that comes to mind is whether the most advanced techniques used in software verification, specifically model checking, can help.

The most used model checking formalisms, however, cannot easily represent constraints that are naturally represented by HSTS, namely continuous time and other continuous parameters. Also, the goal of HSTS is to provide an expressive language to facilitate knowledge acquisition by non AI experts (e.g., system engineers). So, HSTS models cannot be easily translated into a model checking formalism. To allow model checking algorithms to operate on HSTS models we, therefore, need a mapping between a *subset* of the HSTS Domain Description Language to a model checking formalism. An earlier attempt to analyze HSTS planner models, where no metrics were considered, is described in [10].

We choose UPPAAL because it can represent time (section 3), it is comparable to HSTS in terms of representation and search since they are both constraint based systems, and has been successfully applied to several verification cases of real-time systems of industrial interest [4, 3].

Some of the issues that we are interested in addressing in this research are:

1. Whether model checking techniques can address problems of the size of a realistic autonomy planning application;
2. Once we have a mapping from a planner model into a model checking formalism, what the core differences between the search method used in model checking and that used by a planner are; and
3. Lessons, if any, that planning can take from model checking and vice versa regarding representation, search control, and other aspects. Related work can be found in [1, 2].

2 HSTS

HSTS, Heuristic Scheduling Testbed System, is a general constraint-based planner and scheduler. It is also one of four components that constitutes the Remote Agent Autonomous system which was used for the Remote Agent Experiment, (RAX), in May of 1999, for autonomous control of the Deep Space 1 spacecraft [8].

HSTS consists of a planning engine that takes a plan model, in addition to a goal, as input and produces a complete plan. A plan model is a description of the domain given as a set of objects and constraints. The produced plan achieves the specified goal and satisfies the constraints in the plan model.

In more detail, an HSTS plan model is based on a collection of entities, called state variables, that can be in different states, represented as predicate values. Therefore, predicates represent "spans", or intervals, on state variables. A set of compatibilities between predicates is specified. The compatibilities are temporal constraints, which may involve durations, between end points of predicate values. For example, "pred1 meets pred2" indicates that the end of pred1 (certain predicate value) should coincide with the start of pred2 (another predicate value); and "pred1 before[3,5] pred2" indicates that the distance between the end of pred1 and the start of pred2 is between 3 and 5. An HSTS plan is a complete assignment of predicate values for all the state variables that satisfy all compatibilities.

HSTS ensures robustness of schedules by allowing for flexible temporal representations. The quality of generated plans is improved by interleaving planning and scheduling, rather than performing them separately. HSTS also allows natural and efficient handling of concurrent processes, and the modeling language is simple in its uniform representation of actions and states. HSTS has a rich language for expressing temporal relation constraints [9, 5].

3 UPPAAL

UPPAAL, an acronym based on a combination of UPPsala and AALborg universities, is a tool box for modeling, simulation, and verification of real-time systems. The simulator is used for interactive and automated analysis of system behavior during early design stages while the verifier, which is a model-checker, covers the exhaustive dynamic behavior of the system for proving safety and bounded liveness properties. The verifier, which is a symbolic model checker, is implemented using sophisticated constraint-solving techniques where efficiency and optimization are emphasized. Space reduction is accomplished by both local and global reductions. The local reduction involves reducing the amount of space a symbolic state occupies and is accomplished by the compact representation of Difference Bounded Matrix (DBM) for clock constraints. The global reduction involves reducing the number of states to save during a course of reachability analysis [11, 7].

A UPPAAL model consists of a set of timed automata, a set of local clocks, global variables, and synchronizing channels. A node in an automaton may be associated with an invariant, which is a set of clock constraints, for enforcing transitions out of the node. An arc may be associated with a guard for controlling when this transition can be taken. On any transition, local clocks may get reset and global variables may get re-assigned. A trace in UPPAAL is a sequence of states, each of which containing a complete specification of a node from each automata, such that each state is the result of a valid transition from the previous state.

UPPAAL had been proven to be a useful model_checking tool for many domains including distributed multimedia and power controller applications [4, 3].

4 Mapping HSTS models into UPPAAL

Here, we present a sketch of the translation from HSTS plan models into UPPAAL models. Definitions for used terms, and a more formal translation algorithm, are to be provided in the final paper.

Each state variable is represented as a UPPAAL automaton where each value predicate is represented as a node. Transitions of an automaton represent value ordering constraints of the corresponding state variable. Each predicate value is mapped into a main node and two chains of nodes that are of zero duration each. The first chain represents the start of the value span and all its constraints. The second chain represents the end of the value span and all its constraints. Duration constraints are translated into invariants and guards of local clocks. Temporal relation constraints are implemented through communication channels.

A goal in HSTS corresponds to a property in UPPAAL. Similarly, a plan in HSTS corresponds to an execution trace in UPPAAL. More details, with supporting examples, will be given in the final paper.

5 Properties for Verification

UPPAAL allows for verifying properties that are useful for ensuring correctness and detecting inconsistencies and flaws in HSTS plan model. For example, UPPAAL is able to verify the existence of complete plans that satisfy given constraints. It can also detects the mutual exclusion properties of predicate values.

Our experience on a sample problem of a rover that can go from one location to another to collect rocks showed the usefulness, and success, of using UPPAAL for model checking HSTS. More details on this experiment are to be provided in the final paper.

6 Summary

Our work tackles the problem of using Model Checking for the purpose of verifying planning systems.

We have constructed an algorithm that maps plan models into timed automata. The algorithm works well for translating models of limited size and complexity. Since complete constraint planning models are much too complex for a complete translation into a model checking formalism, there is a need for building representative "abstract" models. We will investigate such abstraction in the near future.

After translating a plan model, properties can be checked for detecting inconsistencies and incompleteness in the model. In addition, the model checking search engine can be used as an independent problem solving mechanism for verifying the planning engine. This is possible because goals can be mapped into properties and traces correspond to plans.

We are currently working on identifying a set of verification properties that guarantee a certain degree of coverage for HSTS models and the Planning engine. We are also analyzing the benefits, and limitations, of using a model checker for HSTS verification.

References

[1] A. Cimatti, M. Roveri, and P. Traverso. 1998. Strong planning in non-deterministic domains via model checking. In the Proceedings of the 4th International Conference on Artificial Intelligence Planning System (AIPS98), pp. 36-43. AAAI Press.

[2] M. Di Manzo, E. Giunchiglia, and S. Ruffino. 1998. Planning via model checking in deterministic domains: Preliminary report. In the Proceedings of the 8th International Conference on Artificial Intelligence: Methodology, Systems, and Applications (AIMSA98), pp. 221-229. Springer-Verlag.

[3] K. Havelund, K. G. Larsen, and A. Skou. 1999. Formal Verification of a Power Controller Using the Real-Time Model Checker UPPAAL. In the Proceedings of the 5th International AMAST Workshop on Real-Time and Probabilistic Systems.

[4] K. Havelund, A. Slou, K. G. Larsen, and K. Lund. 1997. Formal Modeling and Analysis of and Audio/Video Protocol: An Industrial Case Study Using UPPAAL. In the Proceedings of the 18th IEEE Real-Time Systems Symposium, pages 14-24. San Francisco, California.

[5] A. K. Jonsson, P. H. Morris, N. Muscettola, and K. Rajan. 1999. Planning in Interplanetary Space: Theory and Practice. American Association for Artificial Intelligence (AAAI-99)

[6] K. G. Larsen, P. Pettersson, and W. Yi. 1997. UPPAAL in a Nutshell In Springer International Journal of Software Tools for Technology Transfer 1(1+2).

[7] K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi. 1997. Efficient Verification of Real-Time Systems: Compact Data Structures and State-Space Reduction. In the Proceedings of the 18th IEEE Real-Time Systems Symposium, pages 14-24. IEEE Computer Society Press.

[8] Muscettola, P. P. Nayak, B. Pell, and B. William. 1998 Remote Agent: To boldly go where no AI system has gone before. Artificial Intelligence 103(1-2):5-48

[9] N. Muscettola. 1994. HSTS: Integrated planning and scheduling. In M. Zweben and M. Fox, eds., Intelligent Scheduling. Morgan Kaufman. 169-212

[10] J. Penix, C. Pecheur, K. Havelund. 1998. Using Model Checking to Validate AI Planner Domain Models. In the Proceedings of the 23rd Annual Software Engineering Workshop, NASA Goddard.

[11] W. Yi, P. Pettersson, and M. Daniels. 1994. Automatic Verification of Real-Time Communicating Systems by Constraint-Solving. In Dieter Hogrefe and Stefan Leue, editors, Proceedings of the 7th International Conference on Formal Description Techniques, pages 223-238. North-Holland.